What can neural networks learn about flow physics?

Tina White
Stanford University
Department of Mechanical Engineering
Stanford, CA 94305
crwhite@stanford.edu

Abstract

The intention of this project is to investigate the suitability of a neural network model as an alternative or extension to methods like reduced order models for the real-time simulation of full Navier-Stokes solutions. The neural network model uses RNNs (Recurrent Neural Nets) made of LSTM (Long Short Term Memory) units to do unsupervised learning of sequences of Computational Fluid Dynamics (CFD) simulations. Methods learned for unsupervised learning of images and videos are applied to CFD datasets, which conceptually have a lot of overlap and can be analyzed using similar techniques. Overall, the RNN models based on LSTMs began to learn representations of the CFD data, predicting the movement of shedding vortices for example. With the right tuning and model setup, the RNN models will likely be able to learn the features of these datasets far more accurately. The next step would be to modify the hyper-parameters of the current model and investigate how the learning can be improved by changing the construction of the model.

1. Introduction

A commercial jet cruises at about 7/10ths the speed of sound. Due to the rotation of the gas turbine engine fan and the increasing velocity of the air over the surface of the fan blades, a portion of the airflow over the fan typically experiences supersonic flow conditions and a weak shock wave. Such a flow field can experience transonic flutter, which results from the unsteady interaction of a structure prone to vibration with a transonic airflow. Phenomena like transonic flutter in turbomachinery are predicted and mitigated using CFD, specifically fluid-structure interaction modeling. Flutter is an aeroelastic issue that can cause near-instantaneous and catastrophic part failure.

A CFD analysis is spatially local and involves computing a partial differential equation on a large number - often millions - of discrete points throughout a computational domain. Although CFD analyses are a vast improvement over traditional methods like wind tunnel or rig testing alone, they are computationally expensive, which is limiting to the engineer trying to rapidly iterate and optimize a product. Optimizing a fan design while mitigating flutter risk can take months because an aeroelastic analysis of a single design typically takes 3-5 days even with modern computational tools. While methods exist to increase the speed of a CFD analysis, no commercially available real-time methods accurately predict the aeroelastic behavior of turbomachinery in the region of transonic stall flutter.

Currently, methods like reduced order modeling are used to compute flow behavior in real-time around a given design. In reduced order modeling (ROM), a dataset already exists with pre-computed CFD simulations. From that dataset, the ROM learns how to make predictions in the future by creating a simple model based on what is already known. The goal of reduced order modelling is to simplify the complex system to run in real time, enabling significant CPU time reductions. The intention of this project is to explore the nature of the information a neural network can learn when given a collection of CFD data. Long term, this is an investigation of the suitability of neural networks as an alternative or an extension to methods like reduced order modeling for the real-time simulation of flows.

Sequence modeling techniques and RNNs could be used predict the state of the flow at future time-steps. The intent of applying an RNN model to CFD data is to identify flow structures and learn to track these structures: vortices, boundary layers, turbulent and laminar flow regions, shear layers, etc. The domain representation using neural networks can be used to create more optimal local bases for use in real-time computation of flow simulations or as a substitute or aid for methods like reduced order modeling. Overall, Neural networks could potentially have many applications to fluid dynamics problems. Computational fluid dynamics datasets conceptually have a lot of overlap with image datasets and there's no reason why they cannot be analyzed with similar techniques.

2. Related Work

This project relies greatly on previous work on video representations by Srivastava et al, 2015 [10] and their code provided on GitHub. [11] Their work uses multilayer Long Short Term Memory (LSTM) networks to learn representations of video sequences. Using the RNN model, they analyze the outputs to see how well the model can extrapolate the learned video representation into the future and into the past.

Applying this method to CFD data may supplement reduced order modeling techniques, which have been applied at Stanford by the Farhat Research Group [12] and applied on a 1D Burger's Equation and an acceleration study of a transport aircraft. The effectiveness of reduced order modeling has already been demonstrated for various CFD and fluid-structure interaction problems including aeroelastic problems [3] and problems involving moving shocks. [8] For domain decomposition, k-means is often used to cluster subdomains [9] and snapshots [1]. In this case, the neural network, instead, will make "sense" of the domain.

One problem encountered consistently in reduced order modeling for CFD is that shock waves are not well modeled by ROBs. A shock wave occurs when a wave moves faster than the local speed of sound. It is characterized by an abrupt, nearly discontinuous change in pressure. Therefore, singular value decomposition (SVD) implementations suffer from unphysical Gibbs' oscillations near the shock wave discontinuity. [4] Capturing the discontinuity and reducing the Gibbs' oscillations is difficult without a full order model in the region of the shock. Some [6] have applied domain decomposition, but they resort to implementing a full order model in the region of the shock in order to resolve it. Neural networks may be less susceptible to this issue.

Also, different forms of domain decomposition are common in fluid mechanics. The domain is often split along mesh lines evenly for parallel computing. Typically the split for a given domain is determined manually by the engineer considering the particulars of a given design. [5] In this case, what makes it possible to compute sub-domains more generally and independent of mesh is that this procedure is not computing full CFD solutions along mesh lines. The results of CFD simulations are givens. ROMs are computed "off the grid." Therefore, there's more freedom in how to create sub-domains, making the application of neural networks for domain decomposition for ROMs full of possibilities.

Artificial neural networks (ANNs) have been proposed to train knowledge bots to identify the idiosyncrasies of CFD simulation software and recognize patterns that can lead to successful simulations. [2] Knowledge bots have been used for applications of CFD, trajectory analysis, and finite-element analysis software. It has been shows that machine learning algorithms can learn the idiosyncrasies of computational simulations and identify regions of instability without giving them information about mathematical form or discretization approaches. [2]

ANNs have also been used to apply aerodynamic pressure loads on unmanned aerial vehicles (UAVs) for the purpose of carrying out finite element (FE) analysis during its structural design process. One way fluid–solid interaction (FSI) for UAV structural design was completed, in which aerodynamics loads obtained from CFD were applied on the vehicle structure for steady-state static FE analysis. Aerodynamic pressure data was sorted in terms of coordinates for different region, and a feed forward back propagation neural network model was trained for each data set to generate approximate pressure functions in terms of coordinates. [7]

After researching the current applications of unsupervised learning to flow physics, it was found that that neural networks have not yet been used to feed-forward a CFD simulation in time using RNNs or LTSMs, which is the intent of this project.

3. Methods

The model in this analysis uses RNNs made of LSTM units to do unsupervised learning according to the modeling procedure of Srivastava et al, 2015. [10] The LSTM unit shown in Figure 1 is the basic building block of the RNN model. Each LSTM unit has a cell which has a state c_t at time t. The cell is like a memory unit. Access to the memory unit for reading or modifying is controlled through sigmoidal gates – the input gate it, forget gate f_t and output gate o_t.

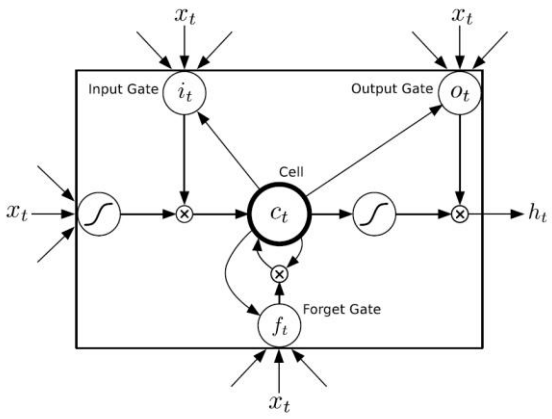

Figure 1: LSTM unit

The model consists of two RNNs – the encoder and the decoder as shown in Figure 2. The state of the encoder after the last input has been read is the representation of the input video. The decoder reconstructs the input sequence from this representation. The representation retains information about the appearance of the objects, the background, and the motion contained in the video.

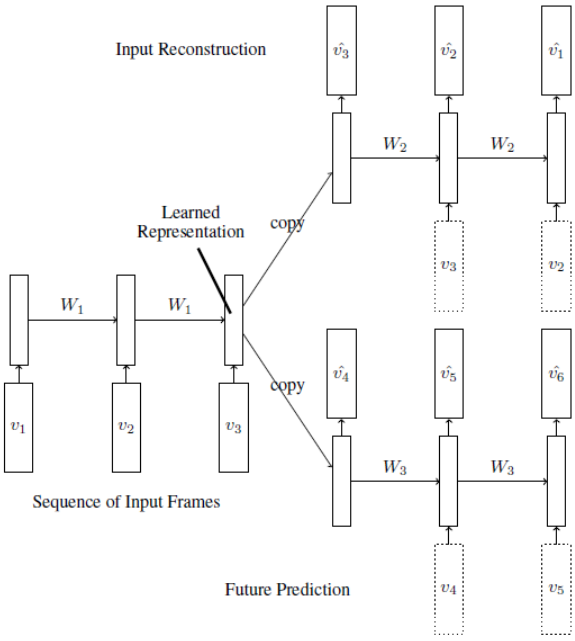

Figure 2: Composite model

The composite model has two tasks – reconstruct the input and predict the future. The encoder comes up with a state from which we can both reconstruct the input and predict the next few frames. The Autoencoder Model consists of two RNNs – the encoder and the decoder. The input to the model is a sequence of image patches. The encoder reads in this sequence. After the last input has been read, the decoder outputs a prediction for the target sequence. The design of the Future Predictor Model is same as that of the Autoencoder Model, except that the decoder in this case predicts a frame sequence that comes after the input sequence. This is the same approach used in language models for modeling sequences of words.

For each dataset, a single layer Composite Model was used. Each LSTM had 2048 units, the encoder took 10 frames as input, the decoder reconstructed these 10 frames, and the future predictor attempted to predict the next 10 frames. Logistic output units were used with a cross entropy loss function. A code implementing this model according to Srivastava et al, 2015 [10] was obtained from GitHub [11].

4. Dataset and Features

4.1 MNIST dataset

The models were first trained on a dataset of moving MNIST digits in order to compare the results of the code to the results obtained by Srivastava et al, 2015. [10] In this dataset, two digits move inside a 64 x 64 patch as shown in Figure 3. The digits were chosen randomly from the training set and placed initially at random locations inside the patch. Each digit was assigned a velocity whose direction was chosen uniformly randomly on a unit circle and whose magnitude was also chosen uniformly at random over a fixed range. The digits bounced-off the edges of the frame. While 10 frames were used in the autoencoder and future predictor, 5 frames of each are plotted in all of the following dataset and results figures.

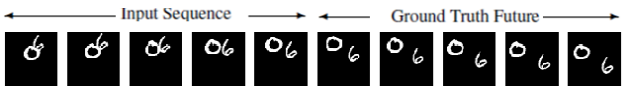

Figure 3: Bouncing MNIST dataset

4.2 Burger's equation dataset

A typical three-dimensional CFD mesh for a transonic stall flutter analysis on a turbomachinery blade contains about 500,000 points. At each point in the mesh, at least 3-5 variables are saved, and the full CFD solution of the fluid motion takes days to complete. Therefore, the first step is to find a simplified problem that can be used for testing for the purposes of this project, but implement it in a way that allows it to be scaled up.

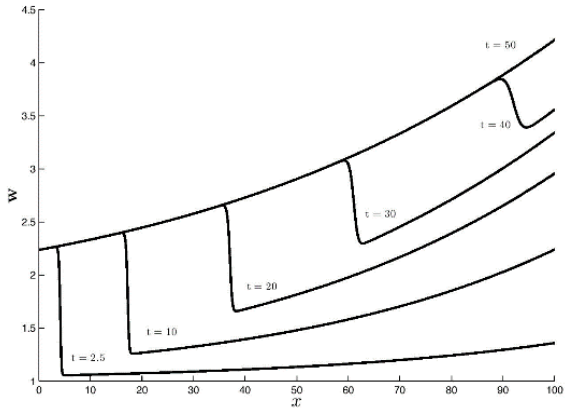

Figure 4: Typical Burger's equation solutions

The solution of a one-dimensional Burgers' problem is such a case. Figure 4 shows simple 1D Burger's equation solutions. It is a one-dimensional fluids application of an initial-boundary-value problem (IBVP) that models the movement of a shockwave. The figure shows the velocity w vs. location x as a shockwave moves from left to right in time across the domain. A small python code can run the one-dimension Burger's problem. An analysis was run with 1000 points in the mesh. One variable, velocity, is saved per coordinate, x. A set of 1000 snapshots was generated. Because the problem is much simpler than a full Navier-Stokes simulation, all of the snapshots were generated in a few minutes. Figure 5 shows a set of snapshots generated for the Burger's equation in time.

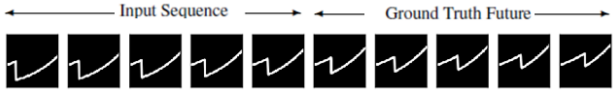

Figure 5: Burger's equation dataset

4.3 CFD dataset

The third dataset was generated using a 2D unsteady computational fluid dynamics solver, which solves the full Navier-Stokes equations. A 2D analysis emulates a slice of a 3D CFD simulation. While the code was developed for airfoils, the physics of a turbomachinery blade is the same. The twisting and flexing of the airfoil emulates the twisting and flexing of a turbomachinery blade undergoing a phenomenon like flutter. Figure 6 shows the response of a fluid to the unsteady motion of a blade at three instances in time.

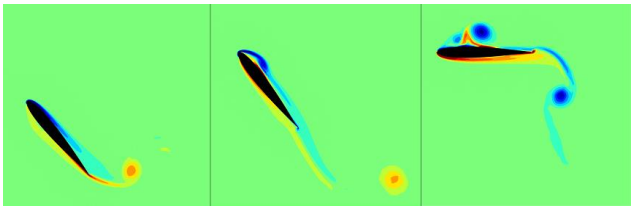
Figure 6: Flow evolution over time as airfoil blade section twists and flexes

This type of simulation is ultimately what the neural network must learn. However, the data vectors are much larger than the Burger's equation vectors. The Burger's dataset was used for debugging, and then the code was trained later on the oscillating airfoil dataset. Figure 7 shows snapshots from a few 2D CFD simulations of an oscillating airfoil.

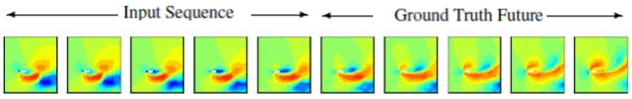

Figure 7: CFD dataset

There were two approaches that could be used to analyze the CFD datasets. In the first approach, the conservation variables computed at each point in the analysis, such as density, momentum, and/or energy, are used in place of RGB channels for image recognition. The data is analyzed directly at mesh points on the CFD mesh. One potential issue with this representation is the warped structure of CFD meshes, see Figure 8, as opposed to the simpler grid structure of images. In the second approach, contours of a computed variable can be plotted to generate images, which are then treated like pixels of visual data for constructing videos. The second approach is used in this study in order to meet the requirement that visual recognition be used for the project, and because it is simpler to implement with current visual recognition methods. However, the second approach is better suited for engineering applications and would produce more meaningful evaluation metrics that can be directly compared to current methods like reduced order modeling. In the second method, the RNN is directly perceiving conservation variables on grid points instead of the more familiar RGB visual images.

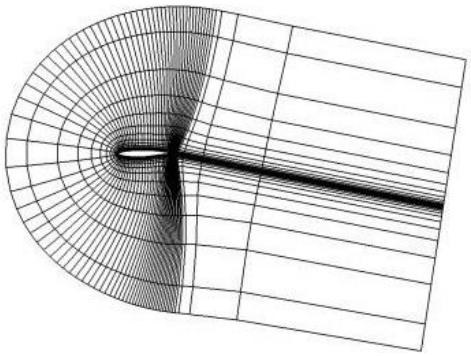

Figure 8: Example of the spatial variation of the grid structure of a C-mesh for a CFD analysis

5. Results

5.1 MNIST results

The MNIST dataset was run to check whether the model was working as intended prior to testing its suitability on the Burger's equation and CFD datasets. Training was run for 1000 iterations before viewing the results. While the results, shown in Figure 9 are not yet sharp predictions, the model is clearly beginning to learn the features of the digits and their motion. This was considered sufficient to move on and begin running the model on the planned datasets.

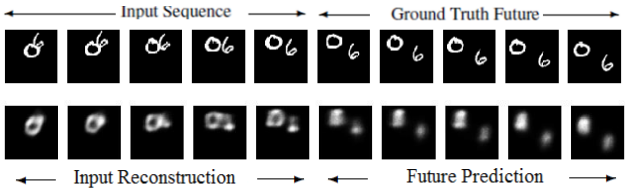

Figure 9: Bouncing MNIST preliminary results

5.2 Burger's equation results

The Burger's equation results are in Figure 10. The model was run for 5000 iterations and, as seen in the MNIST dataset, it is beginning to learn the shockwave features. The issue encountered with the dataset used for training was that the validation set was poorly defined by the training set, and therefore the shockwave dynamics were not well captured. Most of the training set did not include solutions with clear shocks, but simply supersonic solutions that did not exhibit the discontinuity characteristic of shocks. This is why the future predictor predicts future time steps with relatively smooth, flat lines. This is a simple fix, and although there was not enough time to create new training and validation sets for this project, an improved dataset will be generated using the Burger's equation solver.

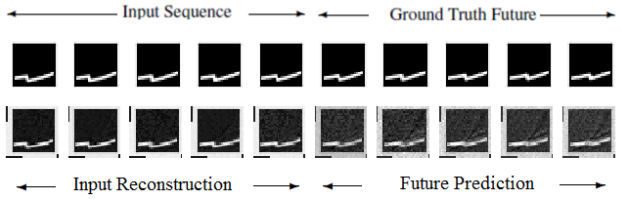

Figure 10: Burger's equation preliminary results

5.3. CFD results

The CFD results are shown in Figures 11 and 12. This model was also run for 5000 iterations. The CFD validation set generated was well represented by the training set, and so even though the dynamics of a full Navier Stokes simulation in 2D are far more complex than the Burger's equation simulations, it is clear that the model was beginning to capture them better. The moving of the shedding vortices, and the pressure changes at the stagnation point at the front of the airfoil are clearly visible. One issue is the apparent noise of the solution – the black specs that dot the images.

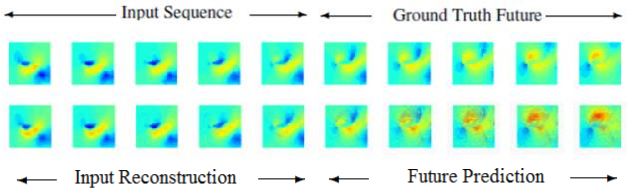

Figure 11: CFD preliminary results

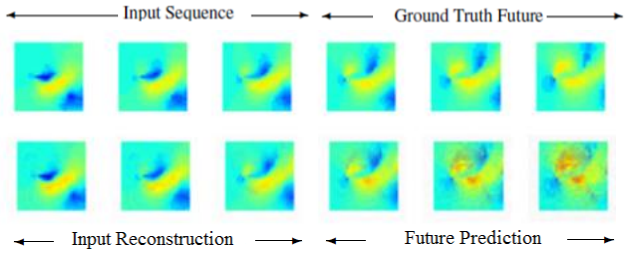

Figure 12: CFD preliminary results full size

Due to project time constraints, only these first preliminary models of neural network solutions to CFD datasets were run. While not highly accurate, the method is clearly able to capture the complexities of some flow phenomenon. At this point, the model will be modified to use the first approach to modeling CFD data, where the conservation variables computed at each point in the

analysis, such as density, momentum, and/or energy, are used in place of RGB channels for image recognition. This will take considerable reorganization of the original code, but will generate results that are more easily comparable to traditional CFD solution methods. They will therefore have more meaningful evaluation metrics than visual inspection.

6. Conclusions / Future Work

The next steps would be to modify the hyper-parameters of the model and investigate the effects of changing the construction of the model. Due to the results of Srivastava et al 2015, [10] it is expected that adding depth to the model will make considerably better predictions, and so the next step is to train a two layer Composite Model, with each layer having 2048 units. Another plan is to change the future predictor by making it conditional in order to make sharper predictions. Another option for improving the model is to apply convolutions across patches of the CFD videos and stack multiple layers of these models. Another investigation would be to check if the model can be tested at time scales that are different from the training time scale.

Visualizing the features learned by this model by looking at the weights that connect each input frame to the encoder will reveal information about what the model is learning about the simulation. Also, the models can be evaluated by looking at the cross entropy of the predictions with respect to the ground truth. As further investigations are completed, these methods can be used to determine their applicability.

Overall, the RNN models based on LSTMs began to learn visually compelling representations of both the shockwave and CFD data. However, at this point the preliminary models are not comparable to alternatives like state-of-the-art reduced order modeling. Currently, the results have been compared and analyzed only through visualization, but the model predictions will be analyzed quantitatively once better models are constructed and trained. Overall, for simplifying and running real-time CFD simulations, neural networks show promise. With the right tuning and model setup, the RNN models will likely be able to learn the representative features of flow physics.

References

[1] Amsallem, David, Matthew J. Zahr, and Charbel Farhat. "Nonlinear model order reduction based on local reduced-order bases." International Journal for Numerical Methods in Engineering 92.10 (2012): 891-916.

[2] Farhat, Charbel, Michael Lesoinne, and P. Le Tallec. "Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity." Computer methods in applied mechanics and engineering 157.1 (1998): 95-114.

[3] Geuzaine, Philippe, et al. "Aeroelastic dynamic analysis of a full F-16 configuration for various flight conditions." AIAA journal 41.3 (2003): 363-371.

[4] Gottlieb, David, and Chi-Wang Shu. "On the Gibbs phenomenon and its resolution." SIAM review 39.4 (1997): 644-668.

[5] Gropp, William D., and David E. Keyes. "Domain decomposition methods in computational fluid dynamics." International journal for numerical methods in fluids 14.2 (1992): 147-165.

[6] Lucia, David J., Paul I. King, and Philip S. Beran. "Domain decomposition for reduced-order modeling of a flow with moving shocks." AIAA journal 40.11 (2002): 2360-2362.

[7] Mazhar, Farrukh, et al. "On using neural networks in UAV structural design for CFD data fitting and classification." Aerospace Science and Technology 30.1 (2013): 210-225.

[8] Samareh, Jamshid A., and Jay Ming Wong. "Training Knowledge Bots for Physics-Based Simulations Using Artificial Neural Networks." (2014).

[9] Smith, Robert E. Computational fluids domain reduction to a simplified fluid network. No. TARDEC-22878. ARMY TANK AUTOMOTIVE RESEARCH DEVELOPMENT AND ENGINEERING CENTER WARREN MI, 2012.

[10] Srivastava, Nitish, Elman Mansimov, and Ruslan Salakhutdinov. "Unsupervised learning of video representations using LSTMs." arXiv preprint arXiv:1502.04681 (2015).

[11] Srivastava, Nitish, P 2015. Unsupervised Learning of Video Representations using LSTMs 1(1):e4, DOI: https://github.com/emansim/unsupervised-videos

[12] Washabaugh, Kyle, et al. "Nonlinear model reduction for CFD problems using local reduced-order bases." 42nd AIAA Fluid Dynamics Conference and Exhibit, Fluid Dynamics and Co-located Conferences, AIAA Paper. Vol. 2686. 2012.